

NESTED DOMAIN DECOMPOSITION WITH POLARIZED TRACES FOR THE 2D HELMHOLTZ EQUATION

LEONARDO ZEPEDA-NÚÑEZ[†] AND LAURENT DEMANET[†]

Abstract. We present a solver for the 2D high-frequency Helmholtz equation in heterogeneous, constant density, acoustic media, with online parallel complexity that scales empirically as $\mathcal{O}(\frac{N}{P})$, where N is the number of volume unknowns, and P is the number of processors, as long as $P = \mathcal{O}(N^{1/5})$. This sublinear scaling is achieved by domain decomposition, not distributed linear algebra, and improves on the $P = \mathcal{O}(N^{1/8})$ scaling reported earlier in [44]. The solver relies on a two-level nested domain decomposition: a layered partition on the outer level, and a further decomposition of each layer in cells at the inner level. The Helmholtz equation is reduced to a surface integral equation (SIE) posed at the interfaces between layers, efficiently solved via a nested version of the polarized traces preconditioner [44]. The favorable complexity is achieved via an efficient application of the integral operators involved in the SIE.

1. Introduction. It has become clear over the past several years that the right mix of ideas to obtain an efficient Helmholtz solver, in the high-frequency regime, involves domain decomposition with accurate transmission conditions. The first $\mathcal{O}(N)$ complexity algorithm, where N is the number of degrees of freedom, was the sweeping preconditioner of Engquist and Ying [14] that uses a decomposition in grid-spacing-thin layers coupled with an efficient multi-frontal solver at each layer. Subsequently, Stolk proposed different instances of domain decomposition methods [38, 39] that restored the ability to use arbitrarily thick layers, improving the efficiency of the local solves at each layer, with the same $\mathcal{O}(N)$ claim. Liu and Ying recently presented a recursive version of the sweeping preconditioner in 3D that decreases the offline cost to linear complexity [28] and a variant of the sweeping preconditioner using an additive Schwarz preconditioner [27]. Many other groups have proposed algorithms with similar complexity claims that we review in Section 1.2.

Although these algorithms are instances of efficient iterative methods, they have to revisit all the degrees of freedom inside the volume in a sequential (or sweeping) fashion at each iteration. This sequential computation thus hinders the scalability of the algorithms to large-scale parallel architectures.

Two solutions have been proposed so far to mitigate the lack of asymptotic scalability in the high-frequency regime:

- Poulson et al. [34] parallelized the sweeping preconditioner in 3D by using distributed linear algebra to solve the local system at each layer, obtaining a $\mathcal{O}(N)$ complexity. This approach should in principle reach sublinear complexity scalings in 2D. The resulting codes are complex, and using distributed linear algebra libraries can be cumbersome for industrial application because of licensing issues.
- Zepeda-Núñez and Demanet [44] proposed the method of polarized traces, with an online runtime $\mathcal{O}(N/P)$ in 2D where N is the number of degrees of freedom in the volume and P is the number of nodes, in a distributed memory environment, provided that $P = \mathcal{O}(N^{1/8})$. The communication cost is a negligible $\mathcal{O}(N^{1/2}P)$. The algorithm is further well-suited to pipelining the right-hand sides.

In this paper we follow, and improve on the latter approach.

[†]Department of Mathematics and Earth Resources Laboratory, Massachusetts Institute of Technology, Cambridge MA 02139, USA. This project is sponsored by Total SA. LD is also funded by NSF, AFOSR, and ONR. Both authors thank Russell Hewett for interesting discussions.

Stage	Polarized traces	Nested polarized traces
offline	$\mathcal{O}(N^{3/2}/P)$	$\mathcal{O}((N/P)^{3/2})$
online	$\mathcal{O}(PN^\alpha + N/P)$	$\mathcal{O}(P^{1-\alpha}N^\alpha + N/P)$

TABLE 1

Runtime of both formulations (up to logarithmic factors), supposing only P nodes, one node per layer for the method of polarized traces, and one node per cell for the nested domain decomposition with polarized traces. Typically $\alpha = 3/4$. In that case, the N/P contribution dominates $P^{1-\alpha}N^\alpha$ as long as $P = O(N^{1/5})$.

The solver has two stages: an expensive but parallel offline stage that is performed only once, and a fast online stage that is performed for each right-hand side (source). The algorithm relies on a layered domain decomposition coupled with a surface integral equation (SIE) that is easy to precondition. The operators involved in the SIE and its preconditioner are precomputed and compressed during the offline stage, leading to the above-mentioned online complexity.

1.1. Results. We propose a variant of the method of polarized traces using a nested domain decomposition approach. This novel approach results in an algorithm with an asymptotic online runtime of $\mathcal{O}(N/P)$ provided that $P = \mathcal{O}(N^{1/5})$, which results in a lower online runtime in a distributed memory environment.

Moreover, the nested polarized traces method also has lower memory footprint¹, and lower offline complexity as shown in Table 1.

Both variants of the polarized traces method are modular; they can easily be extended to more complex physics and higher order discretizations. Finally, they were developed to take advantage of new developments in direct methods such as [41, 21, 35] and in better block-low-rank and butterfly matrix compression techniques such as [4, 26].

In addition, we propose a few minor improvements to the original scheme presented in [44] to obtain better accuracy, and to accelerate the convergence rate. We provide :

1. an equivalent formulation of the method of polarized traces that involves a discretization using Q1 finite elements on a cartesian grid, which is second order accurate despite the roughness of the model, provided that a suitable quadrature rule is used to compute the mass matrix. This formulation can be easily generalized to high-order finite differences and high-order finite elements;
2. and, a variant of the preconditioner introduced in [44], in which we used a block Gauss-Seidel iteration instead of a block Jacobi iteration, that improves the convergence rate.

1.2. Related work. Using domain decomposition to solve the Helmholtz problem was proposed for the first time by Després in [12], which led to the development of many different approaches, in particular, the ultra weak variational formulation (UWVF) [8], which, in return, spawned plane wave methods such as the Trefftz formulation of Perugia et al. [31], the plane wave discontinuous Galerkin method [22, 23], the discontinuous enrichment method of Farhat et al. [18], the partition of unity method (PUM) by Babuska and Melenk [3], the least-squares method by Monk

¹The offline complexity for the method of polarized in this case (Table 1) is higher than in [44] because we assume that we have P nodes instead of $N^{1/2}P$.

and Wang [33], among many others. A recent and thorough review can be found in [24].

The idea of mixing a layered domain decomposition with accurate transmission conditions can be traced back, to a great extent, to the AILU preconditioner of Gander and Nataf [19]. However, it was Engquist and Ying who showed in [14] that such ideas could yield fast methods to solve the high-frequency Helmholtz equation, by introducing the sweeping preconditioner. Since then, many other papers have proposed method with similar claims. Stolk [38] proposed a domain decomposition method using single layer potentials to transfer the information between subdomains. Geuzaine and Vion explored randomized methods to approximate the absorbing boundary conditions via a Dirichlet to Neumann map (DtN) for the transmission of the waves within a multiplicative Schwartz iteration [40]. Chen and Xiang proposed another instance of efficient domain decomposition where the emphasis is on transferring sources from one subdomain to another [10]. Luo et al. proposed the fast sweeping Huygens method, based on an approximate Green's function via geometric optics, coupled with a butterfly algorithm, which can handle transmitted waves in very favorable complexity [29]. Closely related to the content of this paper, we find the method of polarized traces [44], and an earlier version of this work [45].

Alongside iterative methods, some advances have also been made on multigrid methods. Erlangga et al. [16] showed how to implement a simple, although sub-optimal, complex-shifted Laplace preconditioner with multigrid. Another variant of complex-shifted Laplacian method with deflation was studied by Sheikh et al. [37]. The choice of the optimal complex-shift was studied by Cools and Vanroose [11] and by Gander et al. [20]. We point out that most of the multigrid methods mentioned above exhibit a suboptimal dependence of the number of iterations to converge with respect to the frequency, making them ill-suited for high frequency problems. However, they are easy to parallelize as shown by Calandra et al. [7].

A good early review of iterative methods for the Helmholtz equation is in [15]. Another review paper that discussed the difficulties generated by the high-frequency limit is [17].

In another exciting direction, much progress has been made on making direct methods efficient for the Helmholtz equation. Such is the case of Wang et al.'s method [41], which couples multi-frontal elimination with \mathcal{H} -matrices (see [13] for the multi-frontal method). Another example is the work of Gillman, Barnett and Martinsson on computing Dirichlet to Neumann maps in a multiscale fashion [21]. Recently, Ambikasaran et al. [1] proposed a direct solver for acoustic scattering for heterogeneous media using compressed linear algebra to solve an equivalent Lippmann-Schwinger equation. It is not yet clear whether offline linear complexity scalings can be achieved this way, though good direct methods are often faster in practice than the iterative methods mentioned at the beginning of this Section. The main issue with direct methods is the lack of scalability to very large-scale problems due to the memory requirements and prohibitively expensive communication overheads.

Finally, beautiful mathematical reviews of the Helmholtz equation are [32] and [9]. A more systematic and extended exposition of the references mentioned here can be found in [43].

1.3. Organization. The present paper is organized as follows :

- we review briefly the formulation of the Helmholtz problem and the reduction to a boundary integral equation in Section 2;
- in Section 3 we review the method of polarized traces;

- in Section 4 we present two variants of the nested solver, and we provide the empirical complexity observed;
- finally, in Section 5 we present numerical experiments that corroborate the complexity claims.

2. Formulation. In this section we follow mostly [44]. Let Ω be a rectangle in \mathbb{R}^2 , and consider a layered partition of Ω into L slabs, or layers $\{\Omega^\ell\}_{\ell=1}^L$ as shown in Fig 1. Define the squared slowness as $m(\mathbf{x}) = 1/c(\mathbf{x})^2$, $\mathbf{x} = (x, z)$. As in geophysics, we may refer to z as depth, and we suppose that it points downwards. Define the global Helmholtz operator at frequency ω as

$$(2.1) \quad \mathcal{H}u = (-\Delta - m\omega^2)u \quad \text{in } \Omega,$$

with an absorbing boundary condition on $\partial\Omega$ implemented via perfectly matched layers (PML) [5, 25].

Let us define f^ℓ as the restriction of f to Ω^ℓ , i.e., $f^\ell = f\chi_{\Omega^\ell}$. Define the local Helmholtz operators as

$$(2.2) \quad \mathcal{H}^\ell u = (-\Delta - m\omega^2)u \quad \text{in } \Omega^\ell,$$

with absorbing boundary conditions on $\partial\Omega^\ell$. Let u be the solution to $\mathcal{H}u = f$.

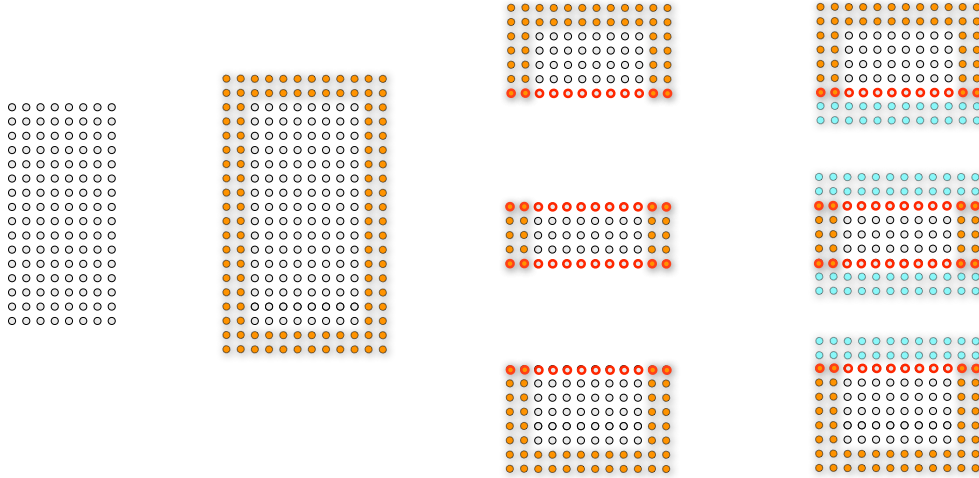


FIG. 1. *Layered domain decomposition.* The orange grid-points represent the PML for the original problem, the light-blue represent the artificial PML between layers, and the red grid-points correspond to $\underline{\mathbf{u}}$ in (2.4).

After discretizing (2.1) using second order finite differences (Appendix A) or Q1 finite elements (Appendix B), we aim to solve the global linear system

$$(2.3) \quad \mathbf{H}\mathbf{u} = \mathbf{f},$$

using a domain decomposition approach that relies on solving the local systems \mathbf{H}^ℓ , which are the discrete version of (2.2).

In Section 2.1 we explain how to reduce (2.3) to an equivalent discrete SIE of the form

$$(2.4) \quad \underline{\mathbf{M}}\underline{\mathbf{u}} = \underline{\mathbf{f}},$$

where \mathbf{u} are the degrees of freedom of \mathbf{u} at the interfaces between layers (see Fig. 1) and \mathbf{M} is defined below.

2.1. Discrete operators. In order to compress the notation, and yet provide enough details so that the reader can implement the algorithm presented in this paper, we recall the notation used in [44].

We suppose that the full domain has $N = n_x \times n_z$ discretization points, that each layer has $n_x \times n^\ell$ discretization points, and that the number of points in the PML, n_{pml} , is the same in both dimension and in every subdomain (light blue and orange nodes in Fig. 1).

In both discretizations the mesh is structured so that we can define $\mathbf{x}_{p,q} = (x_p, z_q) = (ph, qh)$. We assume the same ordering as in [44], i.e.

$$(2.5) \quad \mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{n_z}),$$

and we use the notation

$$(2.6) \quad \mathbf{u}_j = (u_{1,j}, u_{2,j}, \dots, u_{n_x,j}),$$

for the entries of \mathbf{u} sampled at constant depth z . We write \mathbf{u}^ℓ for the wavefield defined locally at the ℓ -th layer, i.e., $\mathbf{u}^\ell = \chi_{\Omega^\ell} \mathbf{u}$, and \mathbf{u}_k^ℓ for the values at the local depth² z_k^ℓ of \mathbf{u}^ℓ . In particular, \mathbf{u}_1^ℓ and $\mathbf{u}_{n^\ell}^\ell$ are the top and bottom rows³ of \mathbf{u}^ℓ . We then gather the interface traces in the vector

$$(2.7) \quad \underline{\mathbf{u}} = (\mathbf{u}_{n^1}^1, \mathbf{u}_1^2, \mathbf{u}_{n^2}^2, \dots, \mathbf{u}_1^{L-1}, \mathbf{u}_{n^{L-1}}^{L-1}, \mathbf{u}_1^L)^t.$$

Define the numerical local Green's function in layer ℓ by

$$(2.8) \quad \mathbf{H}^\ell \mathbf{G}^\ell(\mathbf{x}_{i,j}, \mathbf{x}_{i',j'}) = \mathbf{H}^\ell \mathbf{G}_{i,j,i',j'}^\ell = \delta(\mathbf{x}_{i,j} - \mathbf{x}_{i',j'}),$$

if $(i, j) \in \llbracket -n_{\text{pml}} + 1, n_x + n_{\text{pml}} \rrbracket \times \llbracket -n_{\text{pml}} + 1, n^\ell + n_{\text{pml}} \rrbracket$; where

$$(2.9) \quad \delta(\mathbf{x}_{i,j} - \mathbf{x}_{i',j'}) = \begin{cases} \frac{1}{h^2}, & \text{if } \mathbf{x}_{i,j} = \mathbf{x}_{i',j'}, \\ 0, & \text{if } \mathbf{x}_{i,j} \neq \mathbf{x}_{i',j'}, \end{cases}$$

and where the operator \mathbf{H}^ℓ acts on the (i, j) indices.

Furthermore, for notational convenience we consider \mathbf{G}^ℓ as an operator acting on unknowns at the interfaces, as follows.

DEFINITION 1. We consider $\mathbf{G}^\ell(z_j, z_k)$ as the linear operator defined from $\llbracket -n_{\text{pml}} + 1, n_x + n_{\text{pml}} \rrbracket \times \{z_k\}$ to $\llbracket -n_{\text{pml}} + 1, n_x + n_{\text{pml}} \rrbracket \times \{z_j\}$ given by

$$(2.10) \quad (\mathbf{G}^\ell(z_j, z_k) \mathbf{v})_i = h \sum_{i'=-n_{\text{pml}}+1}^{n_x+n_{\text{pml}}} \mathbf{G}^\ell((x_i, z_j), (x_{i'}, z_k)) \mathbf{v}_{i'},$$

where \mathbf{v} is a vector in $\mathbb{C}^{n_x+2n_{\text{pml}}}$, and $\mathbf{G}^\ell(z_j, z_k)$ are matrices in $\mathbb{C}^{(n_x+2n_{\text{pml}}) \times (n_x+2n_{\text{pml}})}$.

Within this context we define the interface identity operator as

$$(2.11) \quad (\mathbf{Iv})_i = h \mathbf{v}_i.$$

²We hope that there is little risk of confusion in overloading z_j (local indexing) for $z_{n_c^\ell+j}$ (global indexing), where $n_c^\ell = \sum_{j=1}^{\ell-1} n^j$ is the cumulative number of points in depth.

³We do not consider the PML points here.

The interface-to-interface operator $\mathbf{G}^\ell(z_j, z_k)$ is indexed by two depths – following the jargon from fast methods we call them source depth (z_k) and target depth (z_j).

DEFINITION 2. We consider $\mathcal{G}_j^{\uparrow, \ell}(\mathbf{v}_{n^\ell}, \mathbf{v}_{n^\ell+1})$, the up-going local incomplete Green's integral; and $\mathcal{G}_j^{\downarrow, \ell}(\mathbf{v}_0, \mathbf{v}_1)$, the down-going local incomplete Green's integral, as defined by:

$$(2.12) \quad \mathcal{G}_j^{\uparrow, \ell}(\mathbf{v}_{n^\ell}, \mathbf{v}_{n^\ell+1}) = \mathbf{G}^\ell(z_j, z_{n^\ell+1}) \left(\frac{\mathbf{v}_{n^\ell+1} - \mathbf{v}_{n^\ell}}{h} \right) - \left(\frac{\mathbf{G}^\ell(z_j, z_{n^\ell+1}) - \mathbf{G}^\ell(z_j, z_{n^\ell})}{h} \right) \mathbf{v}_{n^\ell+1},$$

$$(2.13) \quad \mathcal{G}_j^{\downarrow, \ell}(\mathbf{v}_0, \mathbf{v}_1) = -\mathbf{G}^\ell(z_j, z_0) \left(\frac{\mathbf{v}_1 - \mathbf{v}_0}{h} \right) + \left(\frac{\mathbf{G}^\ell(z_j, z_1) - \mathbf{G}^\ell(z_j, z_0)}{h} \right) \mathbf{v}_0.$$

In the sequel we use the shorthand notation $\mathbf{G}^\ell(z_j, z_k) = \mathbf{G}_{j,k}^\ell$ when explicitly building the matrix form of the integral systems.

Finally, we define the Newton potential as resulting from a local solve inside each layer.

DEFINITION 3. Consider the local Newton potential \mathcal{N}_k^ℓ applied to a local source \mathbf{f}^ℓ as

$$(2.14) \quad \mathcal{N}_k^\ell \mathbf{f}^\ell = \sum_{j=1}^{n^\ell} \mathbf{G}^\ell(z_k, z_j) \mathbf{f}_j^\ell.$$

By construction $\mathcal{N}^\ell \mathbf{f}^\ell$ satisfies the equation $(\mathbf{H}^\ell \mathcal{N}^\ell \mathbf{f}^\ell)_{i,j} = \mathbf{f}_{i,j}$ for $-n_{pml} + 1 \leq i \leq n_x + n_{pml}$ and $1 \leq j \leq n^\ell$.

Following the notation introduced above, the SIE reduction of the original discrete Helmholtz equation takes the form:

$$(2.15) \quad \mathcal{G}_1^{\downarrow, \ell}(\mathbf{u}_0^\ell, \mathbf{u}_1^\ell) + \mathcal{G}_1^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^\ell, \mathbf{u}_{n^\ell+1}^\ell) + \mathcal{N}_1^\ell \mathbf{f}^\ell = \mathbf{u}_1^\ell,$$

$$(2.16) \quad \mathcal{G}_{n^\ell}^{\downarrow, \ell}(\mathbf{u}_0^\ell, \mathbf{u}_1^\ell) + \mathcal{G}_{n^\ell}^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^\ell, \mathbf{u}_{n^\ell+1}^\ell) + \mathcal{N}_{n^\ell}^\ell \mathbf{f}^\ell = \mathbf{u}_{n^\ell}^\ell,$$

$$(2.17) \quad \mathbf{u}_{n^\ell}^\ell = \mathbf{u}_0^{\ell+1}, \quad \mathbf{u}_{n^\ell+1}^\ell = \mathbf{u}_1^{\ell+1},$$

if $1 < \ell < L$, with

$$(2.18) \quad \mathcal{G}_{n^1}^{\uparrow, 1}(\mathbf{u}_{n^1}^1, \mathbf{u}_{n^1+1}^1) + \mathcal{N}_{n^1}^1 \mathbf{f}^1 = \mathbf{u}_{n^1}^1, \quad \mathbf{u}_{n^1}^1 = \mathbf{u}_0^2, \quad \mathbf{u}_{n^1+1}^1 = \mathbf{u}_1^2,$$

and

$$(2.19) \quad \mathcal{G}_1^{\downarrow, L}(\mathbf{u}_0^L, \mathbf{u}_1^L) + \mathcal{N}_{n^L}^L \mathbf{f}^L = \mathbf{u}_1^L, \quad \mathbf{u}_{n^L-1}^L = \mathbf{u}_0^L, \quad \mathbf{u}_{n^L-1+1}^L = \mathbf{u}_1^L.$$

This was we referred to as $\mathbf{Mu} = \mathbf{f}$ in (2.4).

Finally, the online stage of the algorithm is summarized in Alg. 1.

ALGORITHM 1. Online computation using the SIE reduction

```

1: function  $\mathbf{u} = \text{HELMHOLTZ\_SOLVER}(\mathbf{f})$ 
2:   for  $\ell = 1 : L$  do
3:      $\mathbf{f}^\ell = \mathbf{f}_{\chi_{\Omega^\ell}}$  ▷ partition the source
4:   end for
5:   for  $\ell = 1 : L$  do
6:      $\mathcal{N}^\ell \mathbf{f}^\ell = (\mathbf{H}^\ell)^{-1} \mathbf{f}^\ell$  ▷ solve local problems (parallel)

```

```

7:   end for
8:    $\underline{\mathbf{f}} = (\mathcal{N}_1^1 \mathbf{f}^1, \mathcal{N}_1^2 \mathbf{f}^2, \mathcal{N}_2^2 \mathbf{f}^2, \dots, \mathcal{N}_1^L \mathbf{f}^L)^t$   $\triangleright$  form r.h.s. for the integral system
9:    $\underline{\mathbf{u}} = (\underline{\mathbf{M}})^{-1} \underline{\mathbf{f}}$   $\triangleright$  solve (2.4) for the traces
10:  for  $\ell = 1 : L$  do
11:     $\mathbf{u}_j^\ell = \mathcal{G}_j^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^\ell, \mathbf{u}_{n^\ell+1}^\ell) + \mathcal{G}_j^{\downarrow, \ell}(\mathbf{u}_0^\ell, \mathbf{u}_1^\ell) + \mathcal{N}_j^\ell \mathbf{f}^\ell$   $\triangleright$  reconstruct local solutions
12:  end for
13:   $\mathbf{u} = (\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^{L-1}, \mathbf{u}^L)^t$   $\triangleright$  concatenate the local solutions
14: end function

```

3. The method of polarized traces. In this section, we review succinctly the method of polarized traces; for further details see [43]. From Alg. 1 we can observe that the local solves and the reconstruction can be performed concurrently; the only sequential bottleneck is the solution of (2.4).

The method of polarized traces was developed to solve (2.4) efficiently. The method utilizes another equivalent SIE formulation which relies on :

- a decomposition of the wavefield at the interfaces in two components, up-going and down-going;
- integral relations to close the new extended system;
- a permutation of the unknowns to obtain a easily preconditionable system via matrix splitting.

Following the notation introduced in Section 2.1, the resulting extended system (see Section 3.5 in [44]) can be written as

$$\begin{aligned}
(3.1) \quad & \mathcal{G}_1^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^{\ell, \uparrow}, \mathbf{u}_{n^\ell+1}^{\ell, \uparrow}) + \mathcal{G}_1^{\downarrow, \ell}(\mathbf{u}_0^{\ell, \downarrow}, \mathbf{u}_1^{\ell, \downarrow}) + \mathcal{G}_1^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^{\ell, \downarrow}, \mathbf{u}_{n^\ell+1}^{\ell, \downarrow}) + \mathcal{N}_1^\ell \mathbf{f}^\ell = \mathbf{u}_1^{\ell, \uparrow} + \mathbf{u}_1^{\ell, \downarrow}, \\
(3.2) \quad & \mathcal{G}_{n^\ell}^{\downarrow, \ell}(\mathbf{u}_0^{\ell, \uparrow}, \mathbf{u}_1^{\ell, \uparrow}) + \mathcal{G}_{n^\ell}^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^{\ell, \uparrow}, \mathbf{u}_{n^\ell+1}^{\ell, \uparrow}) + \mathcal{G}_{n^\ell}^{\downarrow, \ell}(\mathbf{u}_0^{\ell, \downarrow}, \mathbf{u}_1^{\ell, \downarrow}) + \mathcal{N}_{n^\ell}^\ell \mathbf{f}^\ell = \mathbf{u}_{n^\ell}^{\ell, \uparrow} + \mathbf{u}_{n^\ell}^{\ell, \downarrow}, \\
(3.3) \quad & \mathcal{G}_0^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^{\ell, \uparrow}, \mathbf{u}_{n^\ell+1}^{\ell, \uparrow}) + \mathcal{G}_0^{\downarrow, \ell}(\mathbf{u}_0^{\ell, \downarrow}, \mathbf{u}_1^{\ell, \downarrow}) + \mathcal{G}_0^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^{\ell, \downarrow}, \mathbf{u}_{n^\ell+1}^{\ell, \downarrow}) + \mathcal{N}_0^\ell \mathbf{f}^\ell = \mathbf{u}_0^{\ell, \uparrow}, \\
(3.4) \quad & \mathcal{G}_{n^\ell+1}^{\downarrow, \ell}(\mathbf{u}_0^{\ell, \uparrow}, \mathbf{u}_1^{\ell, \uparrow}) + \mathcal{G}_{n^\ell+1}^{\uparrow, \ell}(\mathbf{u}_{n^\ell}^{\ell, \uparrow}, \mathbf{u}_{n^\ell+1}^{\ell, \uparrow}) + \mathcal{G}_{n^\ell+1}^{\downarrow, \ell}(\mathbf{u}_0^{\ell, \downarrow}, \mathbf{u}_1^{\ell, \downarrow}) + \mathcal{N}_{n^\ell+1}^\ell \mathbf{f}^\ell = \mathbf{u}_{n^\ell}^{\ell, \downarrow},
\end{aligned}$$

for $\ell = 1, \dots, L$; or equivalently

$$(3.5) \quad \underline{\underline{\mathbf{M}}} \underline{\underline{\mathbf{u}}} = \underline{\underline{\mathbf{f}}}, \quad \underline{\underline{\mathbf{u}}} = \begin{pmatrix} \underline{\mathbf{u}}^\downarrow \\ \underline{\mathbf{u}}^\uparrow \end{pmatrix};$$

where we write

$$(3.6) \quad \underline{\mathbf{u}}^\downarrow = \left(\mathbf{u}_{n^1}^{\downarrow, 1}, \mathbf{u}_{n^1+1}^{\downarrow, 1}, \mathbf{u}_{n^2}^{\downarrow, 2}, \dots, \mathbf{u}_{n^{L-1}}^{\downarrow, L-1}, \mathbf{u}_{n^{L-1}+1}^{\downarrow, L-1} \right)^t,$$

$$(3.7) \quad \underline{\mathbf{u}}^\uparrow = \left(\mathbf{u}_0^{\uparrow, 2}, \mathbf{u}_1^{\uparrow, 2}, \mathbf{u}_0^{\uparrow, 3}, \dots, \mathbf{u}_0^{\uparrow, L}, \mathbf{u}_1^{\uparrow, L} \right)^t,$$

to define the components of the polarized wavefields. The indices and the arrows are chosen such that they reflect the propagation direction. For example, $\mathbf{u}_{n^1}^{\downarrow, \ell}$ represents the wavefield leaving the layer ℓ at its bottom, i.e propagating downwards and sampled at the bottom of the layer.

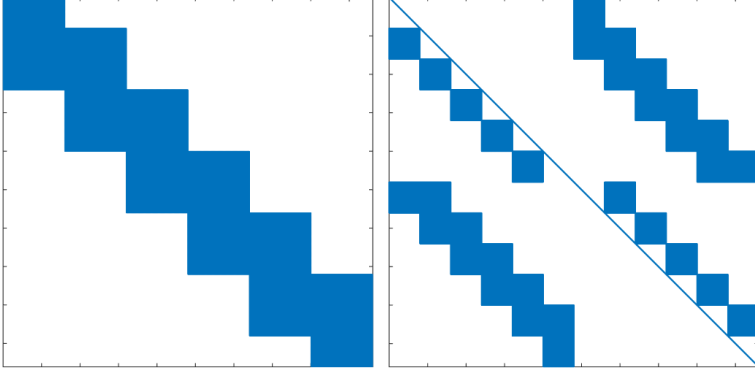


FIG. 2. Sparsity pattern of the SIE matrix in (2.4) (left), and the polarized SIE matrix in (3.8) (right) .

After a permutation of the entries (see Section 3.5 in [44], in particular Fig. 5), and some basic algebraic operations, the matrix in (3.5) takes the form

$$(3.8) \quad \underline{\underline{\mathbf{M}}} = \begin{bmatrix} \underline{\mathbf{D}}^\downarrow & \underline{\mathbf{U}} \\ \underline{\mathbf{L}} & \underline{\mathbf{D}}^\uparrow \end{bmatrix},$$

where $\underline{\mathbf{D}}^\downarrow$ and $\underline{\mathbf{D}}^\uparrow$ are, respectively, block lower and upper diagonal matrices with identity diagonal blocks, thus easily invertible, using a block back-substitution (see Fig. 2 (right)).

Finally the method of polarized traces seeks to solve the system in (3.5) using an iterative method, such as GMRES, coupled with an efficient preconditioner issued from a matrix splitting, which relies on the application of $(\underline{\mathbf{D}}^\downarrow)^{-1}$ and $(\underline{\mathbf{D}}^\uparrow)^{-1}$.

3.1. Gauss-Seidel preconditioner. In this paper, we use a block Gauss-Seidel iteration as a preconditioner to solve the polarized system in (3.5) instead of the block Jacobi iteration used in [44]. The Gauss-Seidel preconditioner is given by

$$(3.9) \quad P^{\text{GS}} \begin{pmatrix} \underline{\mathbf{v}}^\downarrow \\ \underline{\mathbf{v}}^\uparrow \end{pmatrix} = \begin{pmatrix} (\underline{\mathbf{D}}^\downarrow)^{-1} \underline{\mathbf{v}}^\downarrow \\ (\underline{\mathbf{D}}^\uparrow)^{-1} \left(\underline{\mathbf{v}}^\uparrow - \underline{\mathbf{L}}(\underline{\mathbf{D}}^\downarrow)^{-1} \underline{\mathbf{v}}^\downarrow \right) \end{pmatrix},$$

and the Jacobi preconditioner is given by

$$(3.10) \quad P^{\text{Jac}} \begin{pmatrix} \underline{\mathbf{v}}^\downarrow \\ \underline{\mathbf{v}}^\uparrow \end{pmatrix} = \begin{pmatrix} (\underline{\mathbf{D}}^\downarrow)^{-1} \underline{\mathbf{v}}^\downarrow \\ (\underline{\mathbf{D}}^\uparrow)^{-1} \underline{\mathbf{v}}^\uparrow \end{pmatrix},$$

In our experiments, solving (3.8) using GMRES preconditioned with P^{GS} converges twice as fast as using P^{Jac} as a preconditioner, and the former exhibits a weaker dependence of the number of iterations for convergence with respect to the frequency. We considered other standard preconditioners, such as symmetric successive over-relaxation (SSOR) (see section 10.2 in [36]), but they failed to yield faster convergence, while being more computationally expensive to apply.

Fig. 3 depicts the eigenvalues for $\underline{\underline{\mathbf{M}}}$ preconditioned with P^{GS} and P^{Jac} . We can observe that for P^{GS} the eigenvalues are more clustered and there exist fewer outliers. This would explain the fewer number of iteration needed to convergence.

The system in (3.5) is solved using GMRES preconditioned with P^{GS} . Moreover, as in [44] one can use an adaptive \mathcal{H} -matrix fast algorithm for the application of integral kernels.

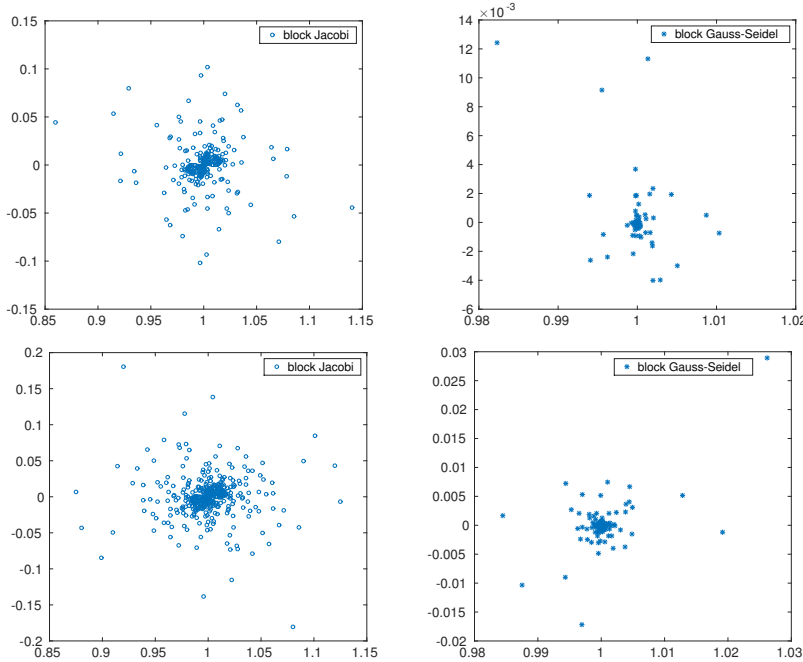


FIG. 3. *Eigenvalues for the preconditioned polarized systems using the block Jacobi (left) and the block Gauss-Seidel (right) preconditioner, using the BP2004 model [6] with $L = 5$, $npml = 10$, and $\omega = 34\pi$ (top row) and $\omega = 70\pi$ (bottom row).*

4. Nested solver. The main drawback of the method of polarized traces is its off-line precomputation that involves computing, storing, and compressing the interface-to-interface Green’s functions needed to assemble $\underline{\underline{\mathbf{M}}}$. In 3D this approach would become impractical given the sheer size of the resulting matrices. To alleviate this issue, we present an equivalent matrix-free approach that relies on local solves with sources at the interfaces between layers.

As it will be explained in the sequel, the matrix-free approach relies on the fact that the blocks of $\underline{\underline{\mathbf{M}}}$ (as well as the blocks of $\underline{\mathbf{D}}^\downarrow$ and $\underline{\mathbf{D}}^\uparrow$) are the restrictions of local Green’s functions. Thus they can be applied via a local solve (using, for example, a direct solver) with sources at the interfaces, which is the same argument used in [44] to reconstruct the solution in the volume (see Section 2.3, in particular Eq. (28), in [44]). However, given the iterative nature of P^{GS} (that relies on inverting $\underline{\mathbf{D}}^\downarrow$ and $\underline{\mathbf{D}}^\uparrow$ by block-backsubstitution) solving the local problems naively would incur a deterioration of the online complexity. This deterioration can be circumvented if we solve the local problems inside the layer via the same boundary integral strategy as in the method of polarized traces, in a nested fashion. This procedure can be written as a factorization of the Green’s integral in block-sparse factors, as will be explained in Section 4.2.

The nested domain decomposition approach involves a layered decomposition in $L \sim \sqrt{P}$ layers, such that each layer is further decomposed in $L_c \sim \sqrt{P}$ cells, as shown in Fig. 4.

In addition to the lower online complexity achieved by the nested approach, the offline complexity is much reduced; instead of computing large Green’s functions for each layer, we compute much smaller interface-to-interface operators between the

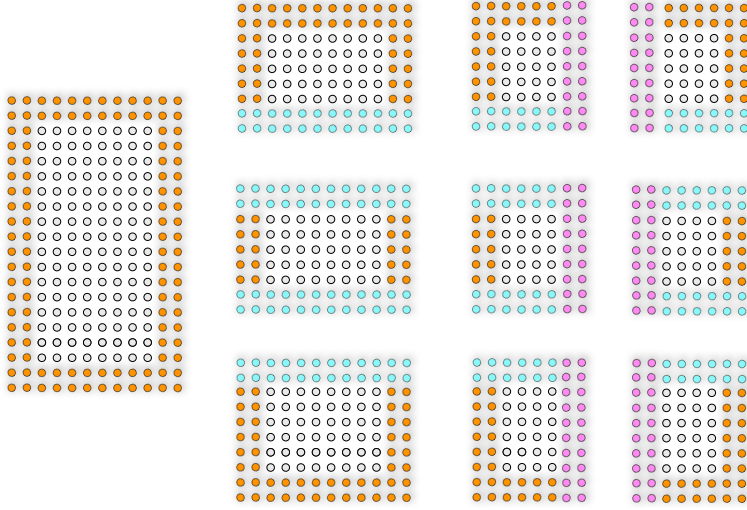


FIG. 4. *Nested Decomposition in cells.* The orange grid-points represent the PML for the original problem, the light-blue represent the artificial PML between layers, and the pink grid-points represent the artificial PML between cells in the same layer. Compare to Fig. 1.

interfaces of adjacent cells within each layer, resulting in a lower memory footprint.

The nested approach consists of two levels:

- the *outer solver*, which solves the global Helmholtz problem, (2.3), using the matrix-free version of the method of polarized traces to solve (2.4) at the interfaces between layers;
- and the *inner solver*, which solves the local Helmholtz problems at each layer, using an integral boundary equation to solve for the degrees of freedom at the interfaces between cells within a layer.

4.1. Matrix-free approach. We proceed to explain how to implement the method of polarized traces using the matrix-free approach. As stated before, the backbone of the method of polarized traces is to solve (3.8) iteratively with GMRES using (3.9) as a preconditioner. Thus, we explain how to apply $\underline{\underline{\mathbf{M}}}$ and P^{GS} in a matrix-free fashion; and we provide the pseudo-code for the method of polarized traces using the matrix-free approach with the local solves explicitly identified.

From (3.1), (3.2), (3.3) and (3.4), each block of $\underline{\underline{\mathbf{M}}}$ is a Green's integral, and its application to a vector is equivalent to sampling a wavefield generated by suitable sources at the boundaries. The application of the Green's integral to a vector $\underline{\mathbf{v}}$, in matrix-free approach, consists of three steps: from $\underline{\mathbf{v}}$ we form the sources at the interfaces, we perform a local direct solve inside the layer, and we sample the solution at the interfaces. The precise algorithm to apply $\underline{\underline{\mathbf{M}}}$ in a matrix-free fashion is provided in Alg. 2.

ALGORITHM 2. *Application of the boundary integral matrix $\underline{\underline{\mathbf{M}}}$*

- 1: **function** $\underline{\mathbf{u}} = \text{BOUNDARY_INTEGRAL}(\underline{\mathbf{v}})$
- 2: $\tilde{\mathbf{f}}^1 = -\delta(z_{n^1+1} - z)\mathbf{v}_{n^\ell}^1 + \delta(z_{n^1} - z)\mathbf{v}_{n^1}^2$
- 3: $\mathbf{w}^1 = (\mathbf{H}^1)^{-1}\tilde{\mathbf{f}}^1$
- 4: $\mathbf{u}_{n^\ell}^\ell = \mathbf{w}_{n^\ell}^\ell - \mathbf{v}_{n^\ell}^\ell$
- 5: **for** $\ell = 2 : L - 1$ **do**

```

6:       $\tilde{\mathbf{f}}^\ell = \delta(z_1 - z)\mathbf{v}_{n^{\ell-1}}^{\ell-1} - \delta(z_0 - z)\mathbf{v}_1^\ell$ 
           $-\delta(z_{n^\ell+1} - z)\mathbf{v}_{n^\ell}^\ell + \delta(z_{n^\ell} - z)\mathbf{v}_1^{\ell+1}$ 
7:       $\mathbf{w}^\ell = (\mathbf{H}^\ell)^{-1}\tilde{\mathbf{f}}^\ell$   $\triangleright$  inner solve
8:       $\mathbf{u}_1^\ell = \mathbf{w}_1^\ell - \mathbf{v}_1^\ell; \quad \mathbf{u}_{n^\ell}^\ell = \mathbf{w}_{n^\ell}^\ell - \mathbf{v}_{n^\ell}^\ell$ 
9:  end for
10:  $\tilde{\mathbf{f}}^L = \delta(z_1 - z)\mathbf{v}_{n^{L-1}}^{L-1} - \delta(z_0 - z)\mathbf{v}_1^L$ 
11:  $\mathbf{w}^L = (\mathbf{H}^L)^{-1}\tilde{\mathbf{f}}^L$ 
12:  $\mathbf{u}_1^L = \mathbf{w}_1^L - \mathbf{v}_1^L$ 
13: end function

```

Alg. 2 can be easily generalized for $\underline{\mathbf{M}}$. We observe that there is no data dependency within the for loop, which yields an embarrassingly parallel algorithm.

Matrix-free preconditioner. For the sake of clarity we present a high level description of the implementation of (3.9) using the matrix-free version.

We use the notation introduced in Section 3 (in particular, (3.6) and (3.7)) to write explicitly the matrix-free operations for the block Gauss-Seidel preconditioner in (3.9). Alg. 3 and 4 have the physical interpretation of propagating the waves across the domains, and Alg. 5 can be seen as the up-going reflections generated by a down-going wave field. The following algorithms can be easily derived from Section 3.5 in [44].

ALGORITHM 3. *Downward sweep, application of $(\underline{\mathbf{D}}^\downarrow)^{-1}$*

```

1: function  $\underline{\mathbf{u}}^\downarrow = \text{DOWNWARD SWEEP}(\underline{\mathbf{v}}^\downarrow)$ 
2:    $\mathbf{u}_{n^1}^{\downarrow,1} = -\mathbf{v}_{n^1}^{\downarrow,1}$   $\triangleright$  invert the diagonal block
3:    $\mathbf{u}_{n^{1+1}}^{\downarrow,1} = -\mathbf{v}_{n^{1+1}}^{\downarrow,1}$ 
4:   for  $\ell = 2 : L - 1$  do
5:      $\mathbf{w}^\ell = (\mathbf{H}^\ell)^{-1} \left[ \delta(z_0 - z)\mathbf{u}_{n^{\ell-1}+1}^{\downarrow,\ell-1} - \delta(z_1 - z)\mathbf{u}_{n^{\ell-1}}^{\downarrow,\ell-1} \right]$   $\triangleright$  inner solve
6:      $\mathbf{u}_{n^\ell}^{\downarrow,\ell} = \mathbf{w}_{n^\ell}^\ell - \mathbf{v}_{n^\ell}^{\downarrow,\ell}$   $\triangleright$  sample the wavefield and subtract the r.h.s.
7:      $\mathbf{u}_{n^{\ell+1}}^{\downarrow,\ell} = \mathbf{w}_{n^{\ell+1}}^\ell - \mathbf{v}_{n^{\ell+1}}^{\downarrow,\ell}$   $\triangleright$  sample the wavefield and subtract the r.h.s.
8:   end for
9:    $\underline{\mathbf{u}}^\downarrow = \left( \mathbf{u}_{n^1}^{\downarrow,1}, \mathbf{u}_{n^{1+1}}^{\downarrow,1}, \mathbf{u}_0^{\downarrow,2}, \dots, \mathbf{u}_0^{\downarrow,L-1}, \mathbf{u}_1^{\downarrow,L-1} \right)^t$ 
10: end function

```

ALGORITHM 4. *Upward sweep, application of $(\underline{\mathbf{D}}^\uparrow)^{-1}$*

```

1: function  $\underline{\mathbf{u}}^\uparrow = \text{UPWARD SWEEP}(\underline{\mathbf{v}}^\uparrow)$ 
2:    $\mathbf{u}_0^{\uparrow,L} = -\mathbf{v}_0^{\uparrow,L}$   $\triangleright$  invert the diagonal block
3:    $\mathbf{u}_1^{\uparrow,L} = -\mathbf{v}_1^{\uparrow,L}$ 
4:   for  $\ell = L - 1 : 2$  do
5:      $\mathbf{w}^\ell = (\mathbf{H}^\ell)^{-1} \left[ -\delta(z_{n^\ell+1} - z)\mathbf{u}_1^{\uparrow,\ell-1} + \delta(z_{n^\ell} - z)\mathbf{u}_0^{\uparrow,\ell-1} \right]$   $\triangleright$  inner solve
6:      $\mathbf{u}_1^{\uparrow,\ell} = \mathbf{w}_1^\ell - \mathbf{v}_1^{\uparrow,\ell}$   $\triangleright$  sample the wavefield and subtract the r.h.s.
7:      $\mathbf{u}_0^{\uparrow,\ell} = \mathbf{w}_0^\ell - \mathbf{v}_0^{\uparrow,\ell}$   $\triangleright$  sample the wavefield and subtract the r.h.s.
8:   end for
9:    $\underline{\mathbf{u}}^\uparrow = \left( \mathbf{u}_0^{\uparrow,1}, \mathbf{u}_1^{\uparrow,1}, \mathbf{u}_{n^2}^{\uparrow,2}, \dots, \mathbf{u}_{n^{L-1}}^{\uparrow,L-1}, \mathbf{u}_{n^{L-1}+1}^{\uparrow,L-1} \right)^t$ 
10: end function

```

ALGORITHM 5. *Upward Reflections, application of $\underline{\mathbf{L}}$*

```

1: function  $\underline{\mathbf{u}}^\uparrow = \text{UPWARD REFLECTIONS}(\underline{\mathbf{v}}^\downarrow)$ 
2:   for  $\ell = 2 : L - 1$  do

```

```

3:       $\mathbf{f}^\ell = \delta(z_1 - z)\mathbf{v}_0^{\downarrow,\ell} - \delta(z_0 - z)\mathbf{v}_1^{\downarrow,\ell}$ 
           $-\delta(z_{n^\ell+1} - z)\mathbf{v}_1^{\downarrow,\ell+1} + \delta(z_{n^\ell} - z)\mathbf{v}_0^{\downarrow,\ell+1}$ 
4:       $\mathbf{w}^\ell = (\mathbf{H}^\ell)^{-1}\mathbf{f}^\ell$   $\triangleright$  inner solve
5:       $\mathbf{u}_1^{\uparrow,\ell} = \mathbf{w}_1^\ell - \mathbf{v}_1^{\downarrow,\ell}$   $\triangleright$  sample the wavefield and subtract the identity
6:       $\mathbf{u}_0^{\uparrow,\ell} = \mathbf{w}_0^\ell$   $\triangleright$  sample the wavefield
7:      end for
8:       $\mathbf{f}^L = \delta(z_1 - z)\mathbf{v}_0^{\uparrow,L} - \delta(z_0 - z)\mathbf{v}_1^{\uparrow,L}$ 
9:       $\mathbf{w}^L = (\mathbf{H}^L)^{-1}\mathbf{f}^L$   $\triangleright$  local solve
10:      $\mathbf{u}_1^{\uparrow,L} = \mathbf{w}_1^L - \mathbf{v}_1^{\downarrow,L}$   $\triangleright$  sample the wavefield and subtract the identity
11:      $\mathbf{u}_0^{\uparrow,L} = \mathbf{w}_0^L$   $\triangleright$  sample the wavefield
12:      $\mathbf{u}^\uparrow = \left( \mathbf{u}_0^{\uparrow,2}, \mathbf{u}_1^{\uparrow,2}, \mathbf{u}_{n^2}^{\uparrow,3}, \dots, \mathbf{u}_{n^{L-1}}^{\uparrow,L-1}, \mathbf{u}_{n^L+1}^{\uparrow,L} \right)^t$ 
13: end function

```

We observe that the for loop in line 2-7 in Alg. 5 is completely parallel. On the other hand, in Alg. 3 and 4, the data dependency within the for loop forces the algorithm to run sequentially. The most expensive operation is the inner solve performed locally at each layer. We will argue in Section 4.2 that using a nested approach, with an appropriate reduction of the degrees of freedom, we can obtain a highly efficient inner solve, which yields a fast application of the preconditioner.

Matrix-free solver. We provide the full algorithm of the matrix-free solver using the method of polarized traces coupled with the Gauss-Seidel preconditioner. The main difference with the original method of polarized traces in [44] is that we use Algs. 2, 3, 4, and 5 to perform the GMRES iteration (line 8 of Alg. 6) instead of compressed matrix-vector multiplications.

ALGORITHM 6. *Matrix-free polarized traces solver*

```

1: function  $\mathbf{u} = \text{MATRIX-FREE SOLVER}(\mathbf{f})$ 
2:   for  $\ell = 1 : L$  do
3:      $\mathbf{f}^\ell = \mathbf{f}_{\chi_{\Omega^\ell}}$   $\triangleright$  partition the source
4:      $\mathbf{w}^\ell = (\mathbf{H}^\ell)^{-1}(\mathbf{f}^\ell)$   $\triangleright$  solve local problems
5:   end for
6:    $\underline{\mathbf{f}} = (\mathbf{w}_{n^1}^1, \mathbf{w}_1^2, \dots, \mathbf{w}_1^L)^t$ ;  $\underline{\mathbf{f}}_0 = (\mathbf{w}_{n^1+1}^1, \mathbf{w}_0^2, \dots, \mathbf{w}_0^L)^t$ 
7:    $\underline{\mathbf{f}} = \begin{pmatrix} \underline{\mathbf{f}} \\ \underline{\mathbf{f}}_0 \end{pmatrix}$   $\triangleright$  form the r.h.s. for the polarized integral system
8:    $\begin{pmatrix} \underline{\mathbf{u}}^\downarrow \\ \underline{\mathbf{u}}^\uparrow \end{pmatrix} = \underline{\mathbf{u}} = (P^{GS}\underline{\mathbf{M}})^{-1} P^{GS}\underline{\mathbf{f}}$   $\triangleright$  solve using GMRES
9:    $\underline{\mathbf{u}} = \underline{\mathbf{u}}^\uparrow + \underline{\mathbf{u}}^\downarrow$   $\triangleright$  add the polarized components
10:   $\tilde{\mathbf{f}}^1 = \mathbf{f}^1 - \delta(z_{n^1+1} - z)\mathbf{u}_{n^1}^1 + \delta(z_{n^1} - z)\mathbf{u}_1^2$   $\triangleright$  reconstruct local solutions
11:   $\mathbf{u}^1 = (\mathbf{H}^1)^{-1}(\tilde{\mathbf{f}}^1)$ 
12:  for  $\ell = 2 : L - 1$  do
13:     $\tilde{\mathbf{f}}^\ell = \mathbf{f}^\ell + \delta(z_1 - z)\mathbf{u}_{n^{\ell-1}}^{\ell-1} - \delta(z_0 - z)\mathbf{u}_1^\ell$ 
           $-\delta(z_{n^\ell+1} - z)\mathbf{u}_{n^\ell}^\ell + \delta(z_{n^\ell} - z)\mathbf{u}_1^{\ell+1}$ 
14:     $\mathbf{u}^\ell = (\mathbf{H}^\ell)^{-1}(\tilde{\mathbf{f}}^\ell)$ 
15:  end for
16:   $\tilde{\mathbf{f}}^L = \mathbf{f}^L + \delta(z_1 - z)\mathbf{u}_{n^{L-1}}^{L-1} - \delta(z_0 - z)\mathbf{u}_1^L$ 
17:   $\mathbf{u}^L = (\mathbf{H}^L)^{-1}(\tilde{\mathbf{f}}^L)$ 
18:   $\mathbf{u} = (\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^{L-1}, \mathbf{u}^L)^t$   $\triangleright$  concatenate the local solutions
19: end function

```

The matrix-free solver in Alg. 6 has three stages :

- *lines 2-7*: preparation of the r.h.s. for the outer polarized integral system;
- *lines 8-9*: solve for the traces at the interfaces between layers, using preconditioned GMRES, and applying $\underline{\mathbf{M}}$ and the preconditioner via the matrix-free approach with Algs. 2, 3, 4 and 5;
- *lines 10-18*: reconstruction of the solution inside the volume at each node.

4.2. Nested inner and outer solver. In the presentation of the matrix free solver (Alg. 6), we have extensively relied on the assumption that the inner systems \mathbf{H}^ℓ can be solved efficiently in order to apply the Green's integrals fast. In this section we describe the algorithms to compute the solutions to the inner systems efficiently, and then we describe how the outer solver calls the inner solver.

From the analysis of the rank of the off-diagonal blocks of the Green's functions we know that the Green's integrals can be compressed in a way that results in a fast application in $\mathcal{O}(n^{3/2})$ time (see Section 5 in [44]), but this approach requires precomputation and storage of the Green's functions. The matrix-free approach in Alg. 6 does not need expensive precomputations, but it would naively perform a direct solve in the volume (inverting \mathbf{H}^ℓ), resulting in an application of the Green's integral in $\mathcal{O}(N/L)$ complexity up to logarithmic factors (assuming that a good direct method is used at each layer). This becomes problematic when applying the preconditioner, which involves $\mathcal{O}(L)$ sequential applications of the Green's integrals as Algs. 3 and 4 show. This means that the unaided application of the preconditioner using the matrix-free approach would result in an algorithm with linear online complexity. The nested strategy (that we present below) mitigates this issue resulting in a lower $\mathcal{O}(L_c(n/L)^{3/2})$ (up to logarithmic factors) complexity for the application of each Green's integral.

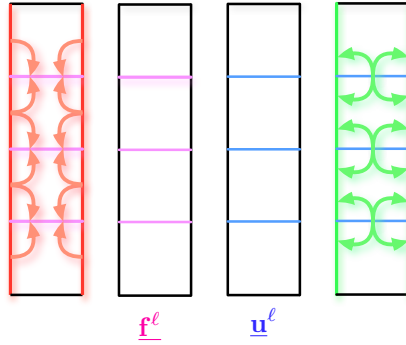


FIG. 5. Sketch of the application of the Green's functions using a nested approach. The sources are in red (left) and the sampled field in green (right). The application uses the inner boundaries as proxies to perform the solve.

We follow the matrix-free approach of Alg. 6, but instead of a direct solver to invert \mathbf{H}^ℓ , we use a nested solver, i.e., we use the same reduction used in Ω to each layer Ω^ℓ . We reduce the local problem at each layer to solving a discrete integral system analog to (2.4) with a layered decomposition in the transverse direction given by

$$(4.1) \quad \underline{\mathbf{M}}^\ell \underline{\mathbf{u}}^\ell = \underline{\mathbf{f}}^\ell, \quad \text{for } \ell = 1, \dots, L_c.$$

We suppose that we have $L_c \sim L \sim \sqrt{P}$ cells in each layer.

The nested solver uses the inner boundaries, or interfaces between cells, as proxies to perform the local solve inside the layer efficiently. The efficiency can be improved when the inner solver is used in the applications of the Green's integral within the preconditioner. In that case, the application of the Green's integral can be decomposed into three steps:

- using precomputed Green's functions at each cell we evaluate the wavefield generated from the sources to form \mathbf{f}^ℓ (from red to pink in Fig. 5 (left)); this operation can be represented by a sparse block matrix $\underline{\mathbf{M}}_f^\ell$;
- we solve (4.1) to obtain $\underline{\mathbf{u}}^\ell$ (from pink to blue in Fig. 5 (right));
- finally, we use the Green's representation formula to sample the wavefield at the interfaces (from blue to green in Fig. 5), this operation is represented by another sparse-block matrix $\underline{\mathbf{M}}_u^\ell$.

Using the definition of the incomplete integrals in Section 2.1 the algorithm described above leads to the factorization

$$(4.2) \quad \begin{bmatrix} \mathcal{G}_0^{\downarrow,\ell}(\mathbf{v}_0, \mathbf{v}_1) + \mathcal{G}_0^{\uparrow,\ell}(\mathbf{v}_{n^\ell}, \mathbf{v}_{n^\ell+1}) \\ \mathcal{G}_1^{\downarrow,\ell}(\mathbf{v}_0, \mathbf{v}_1) + \mathcal{G}_1^{\uparrow,\ell}(\mathbf{v}_{n^\ell}, \mathbf{v}_{n^\ell+1}) \\ \mathcal{G}_2^{\downarrow,\ell}(\mathbf{v}_0, \mathbf{v}_1) + \mathcal{G}_2^{\uparrow,\ell}(\mathbf{v}_{n^\ell}, \mathbf{v}_{n^\ell+1}) \\ \mathcal{G}_{n^\ell+1}^{\downarrow,\ell}(\mathbf{v}_0, \mathbf{v}_1) + \mathcal{G}_{n^\ell+1}^{\uparrow,\ell}(\mathbf{v}_{n^\ell}, \mathbf{v}_{n^\ell+1}) \end{bmatrix} = \underline{\mathbf{M}}_f^\ell \left(\underline{\mathbf{M}}^\ell \right)^{-1} \underline{\mathbf{M}}_u^\ell \cdot \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_{n^\ell} \\ \mathbf{v}_{n^\ell+1} \end{bmatrix},$$

in which the blocks of $\underline{\mathbf{M}}_f^\ell$ and $\underline{\mathbf{M}}_u^\ell$ are dense, but compressible in partitioned low rank (PLR)⁴ form.

Algorithms. We now provide the algorithms in pseudo-code for the the inner solver, Alg. 7, and we provide the necessary modifications to Alg. 6 for the outer solve. In addition, we provide a variant of the inner solver that is crucial to obtain the online complexity mentioned at the beginning of the paper (i.e. $\mathcal{O}(N/P)$ provided that $P = \mathcal{O}(N^{1/5})$).

In order to reduce the notational burden, we define the inner solve using the same notation as before. We suppose that each layer Ω^ℓ is decomposed in L_c cells, noted $\{\Omega^{\ell,c}\}_{c=1}^{L_c}$. We extend all the definitions from the matrix-free solver to the inner solver, by indexing the operations by ℓ and c , in which, ℓ stands for the layer-index and c for the cell-index within the layer.

For each Ω^ℓ , we apply the variable swap $\tilde{\mathbf{x}} = (z, x)$, which is noted by \mathcal{R} such that $\mathcal{R}^2 = I$. Under the variable swap, we can decompose Ω^ℓ in L_c layers $\{\Omega^{\ell,c}\}_{c=1}^{L_c}$ to which we can apply the machinery of the boundary integral reduction at the interfaces between cells. The resulting algorithm has the same structure as before. The variable swap is a suitable tool that allows us to reuse to great extent the notation introduced in [44]. Numerically, the variable swap just introduced is implemented by transposing the matrices that represent the different wavefields.

ALGORITHM 7. *Inner Solve for inverting \mathbf{H}^ℓ in Algs. 3, 4 and 5*

```

1: function  $\mathbf{w} = \text{INNER\_SOLVER}^\ell(\mathbf{f}^\ell)$ 
2:    $\mathbf{g}^\ell = \mathcal{R} \circ \mathbf{f}^\ell$   $\triangleright$  variable swap
3:   for  $c = 1 : L_c$  do
4:      $\mathbf{g}^{\ell,c} = \mathbf{g}\chi_{\Omega^{\ell,c}}$   $\triangleright$  partition the source
5:   end for
6:   for  $c = 1 : L_c$  do
```

⁴A PLR matrix is a \mathcal{H} -matrix obtained by an adaptive dyadic partitioning and multilevel compression, see Section 3 in [44]

```

7:       $\mathcal{N}^{\ell,c} \mathbf{g}^{\ell,c} = (\mathbf{H}^{\ell,c})^{-1} \mathbf{g}^{\ell,c}$   $\triangleright$  solve local problems
8:  end for
9:   $\underline{\mathbf{g}}^\ell = \left( \mathcal{N}_{n^1}^{\ell,1} \mathbf{g}^{\ell,1}, \mathcal{N}_1^{\ell,2} \mathbf{g}^{\ell,2}, \mathcal{N}_{n^2}^{\ell,2} \mathbf{g}^{\ell,2}, \dots, \mathcal{N}_1^{\ell,L_c} \mathbf{g}^{\ell,L_c} \right)^t$   $\triangleright$  form r.h.s.
10:  $\underline{\mathbf{v}}^\ell = \left( \underline{\mathbf{M}}^\ell \right)^{-1} \underline{\mathbf{g}}^\ell$   $\triangleright$  solve for the traces (4.1)
11: for  $c = 1 : L_c$  do  $\triangleright$  local reconstruction
12:    $\mathbf{v}_j^{\ell,c} = \mathcal{G}_j^{\uparrow,\ell,c}(\mathbf{v}_{n^\ell}^{\ell,c}, \mathbf{v}_{n^\ell+1}^{\ell,c}) + \mathcal{G}_j^{\downarrow,\ell,c}(\mathbf{v}_0^{\ell,c}, \mathbf{v}_1^{\ell,c}) + \mathcal{N}_j^{\ell,c} \mathbf{g}^{\ell,c}$ 
13: end for
14:  $\mathbf{v}^\ell = (\mathbf{v}^{\ell,1}, \mathbf{v}^{\ell,2}, \dots, \mathbf{v}^{\ell,L_c-1}, \mathbf{v}^{\ell,L_c})^t$   $\triangleright$  concatenate the local solutions
15:  $\mathbf{w} = \mathcal{R} \circ \mathbf{v}^\ell$   $\triangleright$  variable swap
16: end function

```

To obtain the nested solver, we modify Alg. 6, and the algorithm it calls, by replacing $(\mathbf{H}^\ell)^{-1}$ by the inner solver.

- If the support of the source is the whole layer and the wavefield is required in the volume, we use the inner solve as prescribed in Alg. 7 without modifications. In Alg. 6 we modify lines 4, 11, 14, and 17, in which $(\mathbf{H}^\ell)^{-1}$ is replaced by Alg. 7.
- If the source term is concentrated at the interfaces between layers, and the wavefield is needed only at the interfaces, we reduce the computational cost by using a slight modification of Alg. 7, according to (4.2). In this variant, the local solves in line 7 of Alg. 7 (which is performed via a LU back-substitution) and the reconstruction (lines 11 to 15 in Alg. 7), are replaced by precomputed operators as it was explained above. Within the GMRES loop (line 10 in Alg. 6), we replace $(\mathbf{H}^\ell)^{-1}$ with the variant of Alg. 7 following (4.2), in line 5 of Alg. 3; line 4 of Alg. 4, and lines 4 and 9 of Alg. 5.

The choice of algorithm to solve (4.1) and to apply the Green's integrals dictates the scaling of the offline complexity and the constant of the online complexity. We can either use the method of polarized traces or the compressed-block LU solver, which are explained below.

4.2.1. Nested polarized traces. To efficiently apply the Green's integrals using Alg. 7, we need to solve (4.1) efficiently. One alternative is to use the method of polarized traces in a recursive fashion to solve the system at each layer. We call this approach the method of nested polarized traces. Following [44] this approach has the same empirical scalings, at the inner level, as those found in [44] when the blocks are compressed in PLR form.

Although, as it will be explained Section 4.3, in this case the complexity is lower, we have to iterate inside each layer to solve each system, which produces large constants for the application of the Green's integrals in the online stage.

4.2.2. Inner compressed-block LU. An alternative to efficiently apply the Green's integrals via Alg. 7, is to use the compressed-block LU solver (see Chapter 3 in [43]) to solve (4.1). Given the banded structure of $\underline{\mathbf{M}}^\ell$ (see Fig. 2 (left)), we perform a block LU decomposition without pivoting. The resulting LU factors are block sparse and tightly banded. We have the factorization

$$(4.3) \quad \underline{\mathbf{M}}^\ell = \underline{\mathbf{L}}^\ell \underline{\mathbf{U}}^\ell,$$

which leads to

$$(4.4) \quad \mathcal{G}^\ell = \underline{\mathbf{M}}_f^\ell (\underline{\mathbf{U}}^\ell)^{-1} (\underline{\mathbf{L}}^\ell)^{-1} \underline{\mathbf{M}}_u^\ell,$$

Step	N_{nodes}	Complexity per node	Communication
LU factorizations	$\mathcal{O}(P)$	$\mathcal{O}((N/P + \log(N))^{3/2})$	$\mathcal{O}(1)$
Green's functions	$\mathcal{O}(P)$	$\mathcal{O}((N/P + \log(N))^{3/2})$	$\mathcal{O}(1)$
Local solves	$\mathcal{O}(P)$	$\mathcal{O}(N/P + \log(N)^2)$	$\mathcal{O}(1)$
Sweeps	1	$\mathcal{O}(P(N/P + \log(N)^2)^\alpha)$	$\mathcal{O}(PN^{1/2})$
Recombination	$\mathcal{O}(P)$	$\mathcal{O}(N/P + \log(N)^2)$	$\mathcal{O}(1)$

TABLE 2

Complexity and communication cost of the different steps of the preconditioner, in which α depends on the compression of the local matrices, thus on the scaling of the frequency with respect to the number of unknowns. Typically $\alpha = 3/4$.

in which \mathcal{G}^ℓ represents the linear operator at the left-hand-side of (4.2). Following Section 3 in [43], $(\underline{\mathbf{U}}^\ell)^{-1}(\underline{\mathbf{L}}^\ell)^{-1}$ can be done in the same complexity as the nested polarized traces, at the price of a more thorough precomputation. The improved complexity is achieved by inverting the diagonal blocks of the LU factors, thus reducing $(\underline{\mathbf{U}}^\ell)^{-1}$ and $(\underline{\mathbf{L}}^\ell)^{-1}$ to a sequence of matrix-vector multiplications that are further accelerated by compressing the matrices in PLR form.

The main advantage with respect to using the method of polarized traces in the layer solve, is that we do not need to iterate and the system to solve is half the size. Therefore, the online constants are much lower than using the method of polarized traces for the inner solve.

4.3. Complexity. Table 2 summarizes the complexities and number of processors at each stage of the nested polarized traces method in Section 4.2.1. When using the inner compressed-block LU method, we have the same complexity but with an extra $\mathcal{O}(N^{3/2}/P)$ cost in the offline stage, making it comparable to the method of polarized traces in [44], although with a lower online complexity. For simplicity we do not count the logarithmic factors from the nested dissection; however, we consider the logarithmic factors coming from the extra degrees of freedom in the PML⁵. From Table 2, the offline stage (which is composed of the LU factorizations at each cell containing $\mathcal{O}(N/P + \log(N))$ points, the computation of the Green's function that involves solving $\mathcal{O}(n/\sqrt{P})$ local systems in each cell, and the compression of the resulting Green's functions in PLR form) has an overall runtime $\mathcal{O}((N/P)^{3/2})$, up to logarithmic factors, as stated in Table 1.

For the online stage, the runtime of the local solves and the reconstruction in each cell is independent of the frequency, and it is given by the complexity of multifrontal methods, as stated in Table 2. On the other hand, the runtime of the sweeps depends on the compression ratio of the integral operators, which in return depends on the frequency. If the frequency scales as $\omega \sim \sqrt{n}$, the regime in which second order finite-differences and Q1 finite elements are expected to be accurate, we obtain $\alpha = 5/8$; however, we assume the more conservative value $\alpha = 3/4$. The latter is in better agreement with a theoretical analysis of the rank of the off-diagonal blocks of the Green's functions (see Section 5 in [44]). In such a scenario, we have that the compressed operators $\underline{\mathbf{M}}_u^\ell$ and $\underline{\mathbf{M}}_f^\ell$ (see (4.2)) can be applied in $\mathcal{O}(L_c(n/L + \log(n))^{2\alpha})$ time. In addition, we can solve (4.1) using either the compressed-block LU or the nested polarized traces in $\mathcal{O}(L_c(n/L + \log(n))^{2\alpha})$ time. This yields a runtime of $\mathcal{O}(L_c(n/L + \log(n))^{2\alpha})$ for each application of the Green's integral using

⁵They are more visible in the runtime scalings presented below.

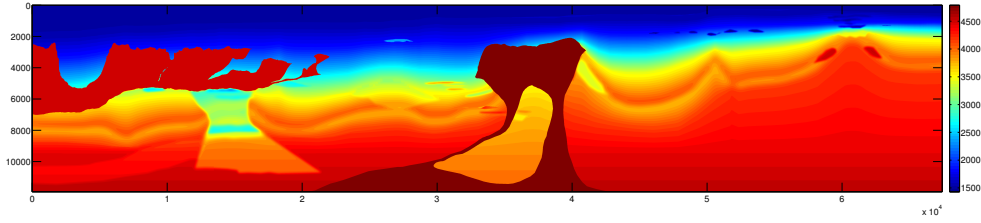


FIG. 6. *BP2004 geophysical benchmark model [6]*

the factorization in (4.2).

To apply the Gauss-Seidel preconditioner we would need $\mathcal{O}(L)$ applications of the Green's integrals, resulting in a runtime of $\mathcal{O}(L \cdot L_c(n/L + \log(n))^{2\alpha})$ to solve (3.5). Using the fact that $L \sim \sqrt{P}$, $L_c \sim \sqrt{P}$, $N = n^2$ and adding the contribution of the other steps of the online stage; we have that the overall online runtime is given by $\mathcal{O}(P^{1-\alpha}N^\alpha + P \log(N)^\alpha + N/P + \log(N)^2)$. Supposing that $P = \mathcal{O}(N)$ and neglecting the logarithmic factors we have that the overall runtime is given by $\mathcal{O}(P^{1-\alpha}N^\alpha + N/P)$ as stated in Table 1.

Moreover, if $\alpha = 3/4$, then we have that the online complexity is $\mathcal{O}(N/P)$ (up to logarithmic factors) provided that $P = \mathcal{O}(N^{1/5})$. The communication cost for the online part is $\mathcal{O}(nP)$, and the memory footprint is $\mathcal{O}(P^{1/4}N^{3/4} + P \log(N)^{3/4} + N/P + \log(N)^2)$, which represents an asymptotic improvement with respect to [44], in which the memory footprint is $\mathcal{O}(PN^{3/4} + N/P + \log(N)^2)$.

5. Numerical results. The code used for the numerical experiments was written in Matlab, and the experiments were generated in a dual socket server with two Xeon E5-2670 and 384 GB of RAM. Fig. 7 depicts the fast convergence of the method when using the BP2004 model (see Fig. 6). After a couple of iterations the exact and approximated solutions are indistinguishable to the naked eye.

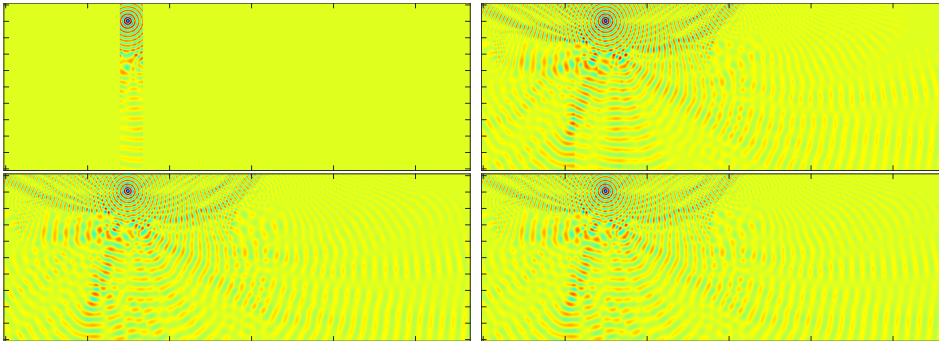


FIG. 7. *Two iteration of the preconditioner, from left to right and top to bottom: initial guess with only local solves; first iteration, second iteration, solution. The background model is given by the BP 2004 model [6] shown in Fig. 6.*

Table 3 shows the sublinear⁶ $\mathcal{O}(P^{1-\alpha}N^\alpha + P \log(N)^\alpha)$ scaling of the runtime of one GMRES iteration for $\alpha = 5/8$, as shown by Fig. 8. We can observe that the number of iterations to converge depends weakly on the frequency and the number of

⁶The same scaling hold for other typical geophysical benchmarks such as Marmousi2 [30], in which convergence is achieved in 4-6 iterations.

N	$\omega/2\pi$ [Hz]	6×2	24×8	42×14	60×20
120×338	2.50	(4) 0.42	(4) 8.30	(4) 24.8	(4) 51.7
239×675	3.56	(4) 0.74	(5) 9.15	(5) 26.1	(5) 52.8
478×1349	5.11	(4) 1.52	(5) 11.6	(5) 30.8	(5) 59.9
955×2697	7.25	(5) 3.32	(5) 17.9	(6) 38.5	(6) 68.8
1910×5394	10.3	(5) 6.79	(6) 29.6	(6) 58.7	(6) 98.3

TABLE 3

Number of GMRES iterations (bold) required to reduce the relative residual to 10^{-5} , along with average execution time (in seconds) of one GMRES iteration using the compressed direct method, for different N and $P = L \times L_c$. The frequency is scaled such that $f = \omega/2\pi \sim \sqrt{n}$, the number of points in the PML scales as $\log(N)$, and the sound speed is given by the BP 2004 model (see [6]).

subdomains. From Fig. 8, we can observe that both methods (nested polarized traces and compressed-block LU) have the same asymptotic runtime, but with different constants⁷.

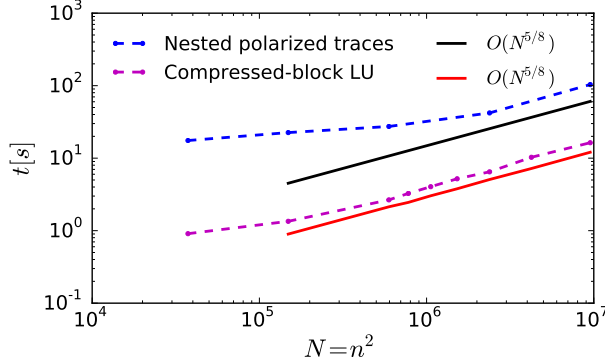


FIG. 8. Runtime for one GMRES iteration using the two different nested solves, for $L = 9$ and $L_c = 3$, $\omega \sim \sqrt{n}$, in which the maximum ϵ -rank in the adaptive PLR compression (see Section 5.1 in [44] for further details) scales as $\max_{\text{rank}} \epsilon \sim \sqrt{\omega}$ and $\epsilon = 10^{-8}$. Moreover, for the nested polarized traces, the accuracy for the GMRES inner solve is fixed to 10^{-6} .

6. Conclusion. We presented an extension to the method of polarized traces introduced in [44], with improved asymptotic runtimes in a distributed memory environment. The method has sublinear runtime even in the presence of rough media of geophysical interest. Moreover, its performance is completely agnostic to the source.

The method can be embedded efficiently within algorithms that require to solve systems in which the medium is locally updated in an inversion loop. The method only needs to be locally modified in order to solve the updated system, thus reducing the overall computational effort. This algorithm is of special interest in the context of time-lapse full-waveform inversion, continuum reservoir monitoring, and local inversion.

We point out that this approach can be further parallelized using distributed algebra libraries. Moreover, the sweeps can be pipelined to maintain a constant load among all the nodes.

Appendix A. Discretization using finite differences.

In order to impose the absorbing boundary conditions, we extend the rectangular

⁷We point out that some gains can be made by using different compressed operators. One can use one compressed operator with high accuracy to apply $\underline{\mathbf{M}}$, an operation that is easily parallelizable, and another with low accuracy to apply the preconditioner that represents the sequential bottleneck.

domain $\Omega = (0, L_x) \times (0, L_z)$ to $\Omega^{\text{ext}} = (-\delta_{\text{pml}}, L_x + \delta_{\text{pml}}) \times (-\delta_{\text{pml}}, L_z + \delta_{\text{pml}})$. The Helmholtz operator in (2.1) then takes the form

$$(A.1) \quad \mathcal{H} = -\partial_{xx} - \partial_{zz} - m\omega^2, \quad \text{in } \Omega^{\text{ext}},$$

where m is an extension⁸ of the squared slowness. The differential operators are redefined following

$$(A.2) \quad \partial_x \rightarrow \alpha_x(\mathbf{x})\partial_x, \quad \partial_z \rightarrow \alpha_z(\mathbf{x})\partial_z,$$

where

$$(A.3) \quad \alpha_x(\mathbf{x}) = \frac{1}{1 + i \frac{\sigma_x(\mathbf{x})}{\omega}}, \quad \alpha_z(\mathbf{x}) = \frac{1}{1 + i \frac{\sigma_z(\mathbf{x})}{\omega}}.$$

Moreover, $\sigma_x(\mathbf{x})$ is defined as

$$(A.4) \quad \sigma_x(\mathbf{x}) = \begin{cases} \frac{C}{\delta_{\text{pml}}} \left(\frac{x}{\delta_{\text{pml}}} \right)^2, & \text{if } x \in (-\delta_{\text{pml}}, 0), \\ 0, & \text{if } x \in [0, L_x], \\ \frac{C}{\delta_{\text{pml}}} \left(\frac{x-L_x}{\delta_{\text{pml}}} \right)^2, & \text{if } x \in (L_x, L_x + \delta_{\text{pml}}), \end{cases}$$

and similarly for $\sigma_z(\mathbf{x})$ ⁹. In general, δ_{pml} goes from a couple of wavelengths in a uniform medium, to a large number independent of ω in a highly heterogeneous medium; and C is chosen to provide enough absorption.

With this notation we rewrite (2.1) as

$$(A.5) \quad \mathcal{H}u = f, \quad \text{in } \Omega^{\text{ext}},$$

with homogeneous Dirichlet boundary conditions (f is the zero extended version of f to Ω^{ext}).

We discretize Ω as an equispaced regular grid of stepsize h , and of dimensions $n_x \times n_z$. For the extended domain Ω^{ext} , we extend this grid by $n_{\text{pml}} = \delta_{\text{pml}}/h$ points in each direction, obtaining a grid of size $(2n_{\text{pml}} + n_x) \times (2n_{\text{pml}} + n_z)$. Define $\mathbf{x}_{p,q} = (x_p, z_q) = (ph, qh)$.

We use the 5-point stencil Laplacian to discretize (A.5). For the interior points $\mathbf{x}_{i,j} \in \Omega$, we have

$$(A.6) \quad (\mathbf{H}u)_{p,q} = -\frac{1}{h^2} (\mathbf{u}_{p-1,q} - 2\mathbf{u}_{p,q} + \mathbf{u}_{p+1,q}) - \frac{1}{h^2} (\mathbf{u}_{p,q-1} - 2\mathbf{u}_{p,q} + \mathbf{u}_{p,q+1}) - \omega^2 m(\mathbf{x}_{p,q}).$$

In the PML, we discretize $\alpha_x \partial_x (\alpha_x \partial_x u)$ as

$$(A.7) \quad \alpha_x(\mathbf{x}_{p,q}) \frac{\alpha_x(\mathbf{x}_{p+1/2,q})(\mathbf{u}_{p+1,q} - \mathbf{u}_{p,q}) - \alpha_x(\mathbf{x}_{p-1/2,q})(\mathbf{u}_{p,q} - \mathbf{u}_{p-1,q})}{h^2},$$

and analogously for $\alpha_z \partial_z (\alpha_z \partial_z u)$. Although we use a second order finite difference stencil, the method can be easily extended to higher order finite differences. In such cases, the number of traces needed in the SIE reduction will increase.

Appendix B. Discretization using Q1 finite elements. The symmetric formulation of the Helmholtz equation takes the form

$$(B.1) \quad - \left(\nabla \cdot \Lambda \nabla + \frac{\omega^2 m(\mathbf{x})}{\alpha_x(\mathbf{x}) \alpha_z(\mathbf{x})} \right) u(\mathbf{x}) = \frac{f(\mathbf{x})}{\alpha_x(\mathbf{x}) \alpha_z(\mathbf{x})},$$

⁸We assume that $m(x)$ is given to us in Ω^{ext} .

⁹In practice, δ_{pml} and C can be seen as parameters to be tuned for accuracy versus efficiency.

where

$$(B.2) \quad \Lambda(\mathbf{x}) = \begin{bmatrix} s_x(\mathbf{x}) & 0 \\ 0 & s_z(\mathbf{x}) \end{bmatrix},$$

$s_x = \alpha_x/\alpha_z$, $s_z = \alpha_z/\alpha_x$, where α_x and α_z are defined in (A.3).

In the case of a medium with sharp interfaces, finite differences approximations give inaccurate results due to the lack of differentiability of the velocity profile. In such cases, sophisticated quadratures and adaptive meshes have to be implemented to properly approximate the finite difference operator [42, 2]. We opted for a low order Q1 finite element discretization, with an adaptive quadrature rule at the discontinuities.

(B.1) is discretized using Q1 elements, leading to a discretized matrix

$$(B.3) \quad \mathbf{H} = \mathbf{S} - \mathbf{M},$$

where the stiffness matrix \mathbf{S} is computed using a Gauss quadrature. On the other hand the mass matrix, \mathbf{M} , is computed using a quadrature adapted to each element depending on the local smoothness of the velocity profile:

- if the medium is locally smooth, a fixed Gauss quadrature is used to approximate the integral over the square;
- if the medium is discontinuous, an adaptive trapezoidal rule is used, until a preset accuracy is achieved.

To discriminate whether the medium is discontinuous, the velocity is sampled at the Gauss-points, and the ratio between maximum and minimum velocity is computed. If the ratio is smaller than a fixed threshold, the medium is considered smooth; otherwise it is considered discontinuous.

Using a nodal basis we can write the system to solve as

$$(B.4) \quad \mathbf{H}\mathbf{u} = \mathbf{f},$$

where \mathbf{u} is the point-wise value of the solution at the corners of the mesh and \mathbf{f} is the projection of f onto the Q1 elements, using a high-order quadrature rule.

The discretization is second order accurate even in the case of discontinuous interfaces with sharp contrasts, as long as the adaptive quadrature rule is used to ensure a small error on the numerical integration.

Appendix C. Green's representation formula.

We present a generalization of the domain decomposition framework developed in [44] to Q1 regular finite elements.

We start by providing the algebraic formula for the discrete Green's representation formula. We propose a technique to derive such formulas without the time consuming computations performed in the Appendix of [44]. We point out that there are clear parallels between this derivation and the reduction to an interface problem using interior Schur complements. However, for the interface system based on Schur complements, we could not define the polarizing conditions that would allow us to construct an efficient preconditioner, forcing us to solve it using a non-scalable direct solver.

In the sequel, we perform extensive manipulations on the matrix \mathbf{H} , thus we introduce some notation to help the reader follow the computations. Following that

ordering define in Section 2.1 we write \mathbf{H} (and \mathbf{H}^ℓ) as a block matrix in the form

$$(C.1) \quad \mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,1} & \mathbf{H}_{1,2} & & & \\ \mathbf{H}_{2,1} & \mathbf{H}_{2,2} & \mathbf{H}_{2,3} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \mathbf{H}_{n_z-1,n_z} \\ & & & \mathbf{H}_{n_z,n_z-1} & \mathbf{H}_{n_z,n_z} \end{bmatrix},$$

in which each block correspond to a fixed z .

We want to derive the algebraic form of the Green's representation formula. From Theorem 1 in [44] we know that using the Green's representation formula locally in a subdomain would produce a discontinuous solution, such that the exact solution is recovered inside the domain, and it is zero outside it. The rationale behind the formalism presented in this section is to find the form of the forcing terms necessary to force the discontinuity of the local representation¹⁰.

An easy manner to deduce the Green's representation formula is to let

$$(C.2) \quad \mathbf{v}^\ell = \mathbf{u}\chi_{\Omega^\ell},$$

which is discontinuous, and apply the local differential operator to \mathbf{v}^ℓ . Finding the discrete Green's representation formula can be re-cast as finding the expression of a system of the form

$$(C.3) \quad \mathbf{H}^\ell \mathbf{v}^\ell = \mathbf{f}^\ell + \mathcal{F}^\ell(\mathbf{u}),$$

such that its solution \mathbf{v}^ℓ satisfies $\mathbf{v}^\ell = \mathbf{u}\chi_{\Omega^\ell}$, and \mathcal{F} depends on the global wavefield \mathbf{u} . In (C.3) we suppose that $\mathbf{f}^\ell = \mathbf{f}\chi_{\Omega^\ell}$ and that \mathbf{H} and \mathbf{H}^ℓ coincide exactly inside the layer. Within this context the problem of finding the formula for the Green's representation formula can be reduced to finding the expression of $\mathcal{F}^\ell(\mathbf{u})$ such that \mathbf{v}^ℓ satisfies Eq C.2.

For ℓ fixed we can obtain the expression of \mathcal{F}^ℓ by evaluating (C.3) and imposing that $\mathbf{v}^\ell = \mathbf{u}\chi_{\Omega^\ell}$. In particular, we need to evaluate (C.3) at the interior of the slab, at its boundaries and at the exterior.

At the interior of the slab $\mathcal{F}^\ell(\mathbf{u})$ is zero, because \mathbf{v}^ℓ satisfies $\mathbf{H}^\ell \mathbf{v}^\ell = \mathbf{H}\mathbf{u} = \mathbf{f} = \mathbf{f}^\ell$.

At the boundaries, the situation is slightly more complex. If we evaluate (C.3) at $k = 1$, we have that

$$(C.4) \quad \mathbf{H}_{1,1}^\ell \mathbf{v}_1^\ell + \mathbf{H}_{1,2}^\ell \mathbf{v}_2^\ell = \mathbf{f}_1^\ell + \mathcal{F}_1(\mathbf{u}).$$

Moreover, evaluating $\mathbf{H}\mathbf{u} = \mathbf{f}$ at the same index yields

$$(C.5) \quad \mathbf{H}_{1,0}\mathbf{u}_0 + \mathbf{H}_{1,1}\mathbf{u}_1 + \mathbf{H}_{1,2}\mathbf{u}_2 = \mathbf{f}_1^\ell.$$

By imposing that $\mathbf{v}^\ell = \mathbf{u}\chi_{\Omega^\ell}$ and subtracting (C.4) and (C.5), we have that

$$(C.6) \quad \mathcal{F}_1^\ell(\mathbf{u}) = -\mathbf{H}_{1,0}\mathbf{u}_0 = -\mathbf{H}_{1,0}^\ell \mathbf{u}_0.$$

We can observe that the role of \mathcal{F}^ℓ is to complete (C.3) at the boundary with exterior data, such that \mathbf{v}^ℓ satisfies the same equation that \mathbf{u} inside the whole layer and not only in the interior.

¹⁰The technique to compute the Green's representation formula, and therefore the transmission operators in form of an incomplete Green's integral, was first mentioned, to the authors knowledge, in Appendix 3B in [43]. More recently, an analogous formulation was used in [39] (see Eqs. (11), (12) and (13)).

Analogously (C.3) can be evaluated at $k = 0$ obtaining

$$(C.7) \quad \mathbf{H}_{0,1}^\ell \mathbf{v}_1^\ell = \mathcal{F}_0^\ell(\mathbf{u}),$$

and imposing that $\mathbf{v}^\ell = \mathbf{u} \chi_{\Omega^\ell}$ we obtain that

$$(C.8) \quad \mathcal{F}_0^\ell(\mathbf{u}) = \mathbf{H}_{0,1} \mathbf{u}_1 = \mathbf{H}_{0,1}^\ell \mathbf{u}_1.$$

Finally, for $k < 0$, the same argument leads to

$$(C.9) \quad \mathcal{F}_k^\ell(\mathbf{u}) = 0.$$

We can easily generalize this argument for the other side of a layer obtaining a generic formula for \mathcal{F}^ℓ

$$\mathcal{F}^\ell(\mathbf{u}) = -\delta_{n^\ell} \mathbf{H}_{n^\ell, n^\ell+1}^\ell \mathbf{u}_{n^\ell+1} + \delta_{n^\ell+1} \mathbf{H}_{n^\ell+1, n^\ell}^\ell \mathbf{u}_{n^\ell} - \delta_1 \mathbf{H}_{1,0}^\ell \mathbf{u}_0 + \delta_0 \mathbf{H}_{0,1}^\ell \mathbf{u}_1,$$

which can be substituted in (C.3), leading to

$$(C.10) \quad \mathbf{H}^\ell \mathbf{v}^\ell = -\delta_{n^\ell} \mathbf{H}_{n^\ell, n^\ell+1}^\ell \mathbf{u}_{n^\ell+1} + \delta_{n^\ell+1} \mathbf{H}_{n^\ell+1, n^\ell}^\ell \mathbf{u}_{n^\ell} - \delta_1 \mathbf{H}_{1,0}^\ell \mathbf{u}_0 + \delta_0 \mathbf{H}_{0,1}^\ell \mathbf{u}_1 + \mathbf{f}^\ell.$$

In addition, (C.10) can be transformed into the discrete expression of the Green's representation formula by applying the inverse of \mathbf{H}^ℓ , \mathbf{G}^ℓ . We can then reformulate the Green's integral in Def. 2 in the form

$$(C.11) \quad \mathcal{G}_j^{\downarrow, \ell}(\mathbf{v}_0, \mathbf{v}_1) = h \begin{bmatrix} \mathbf{G}^\ell(z_j, z_1) & \mathbf{G}^\ell(z_j, z_0) \end{bmatrix} \begin{pmatrix} -\mathbf{H}_{1,0}^\ell \mathbf{v}_0 \\ \mathbf{H}_{0,1}^\ell \mathbf{v}_1 \end{pmatrix},$$

$$(C.12) \quad \mathcal{G}_j^{\uparrow, \ell}(\mathbf{v}_{n^\ell}, \mathbf{v}_{n^\ell+1}) = h \begin{bmatrix} \mathbf{G}^\ell(z_j, z_{n^\ell+1}) & \mathbf{G}^\ell(z_j, z_{n^\ell}) \end{bmatrix} \begin{pmatrix} \mathbf{H}_{n^\ell+1, n^\ell}^\ell \mathbf{v}_{n^\ell} \\ -\mathbf{H}_{n^\ell, n^\ell+1}^\ell \mathbf{v}_{n^\ell+1} \end{pmatrix}.$$

Finally, we can redefine $\mathbf{G}^\ell(z_j, z_k)$ for $k = 0, 1, n^\ell, n^\ell + 1$, such that they absorb all the extra factors. In particular, we redefine:

$$(C.13) \quad \mathbf{G}_{1,0}^\ell = -\delta_1 ((\mathbf{H}^\ell)^{-1} \delta_0 \mathbf{H}_{1,0}^\ell), \quad \mathbf{G}_{1,1}^\ell = -\delta_1 ((\mathbf{H}^\ell)^{-1} \delta_1 \mathbf{H}_{0,1}^\ell),$$

$$(C.14) \quad \mathbf{G}_{n,n}^\ell = -\delta_n ((\mathbf{H}^\ell)^{-1} \delta_{n+1} \mathbf{H}_{n+1,n}^\ell), \quad \mathbf{G}_{n,n+1}^\ell = -\delta_n ((\mathbf{H}^\ell)^{-1} \delta_n \mathbf{H}_{n,n+1}^\ell).$$

This redefinition allows us to seamlessly use all the machinery introduced in [44] to define the SIE and its preconditioner.

REMARK 1. *As an example, in the case of the unsymmetric finite difference discretization, the upper and lower diagonal blocks of \mathbf{H} are diagonal matrices rescaled by $-1/h^2$. Then the formula presented here reduces exactly to the formulas computed by summation by parts in Appendix of [44].*

REFERENCES

- [1] S. Ambikasaran, C. Borges, L.-M. Imbert-Gerard, and L. Greengard. Fast, adaptive, high order accurate discretization of the Lippmann-Schwinger equation in two dimension. *ArXiv e-prints*, [math.NA] 1505.07157, 2015.
- [2] I. Babuska, F. Ihlenburg, E. T. Paik, and S. A. Sauter. A generalized finite element method for solving the Helmholtz equation in two dimensions with minimal pollution. *Computer Methods in Applied Mechanics and Engineering*, 128(3-4):325–359, 1995.

- [3] I. Babuska and J. M. Melenk. The partition of unity method. *International Journal for Numerical Methods in Engineering*, 40(4):727–758, 1997.
- [4] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, volume 63 of *Lecture Notes in Computational Science and Engineering (LNCSE)*. Springer-Verlag, 2008. ISBN 978-3-540-77146-3.
- [5] J.-P. Bérenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114(2):185–200, 1994.
- [6] F. Billette and S. Brandsberg-Dahl. *The 2004 BP velocity benchmark*. EAGE, 2005.
- [7] H. Calandra, S. Gratton, X. Pinel, and X. Vasseur. An improved two-grid preconditioner for the solution of three-dimensional Helmholtz problems in heterogeneous media. *Numerical Linear Algebra with Applications*, 20(4):663–688, 2013.
- [8] O. Cessenat and B. Després. Using plane waves as base functions for solving time harmonic equations with the ultra weak variational formulation. *Journal of Computational Acoustics*, 11(02):227–238, 2003.
- [9] S. N. Chandler-Wilde, I. G. Graham, S. Langdon, and E. A. Spence. Numerical-asymptotic boundary integral methods in high-frequency acoustic scattering. *Acta Numerica*, 21:89–305, 5 2012.
- [10] Z. Chen and X. Xiang. A source transfer domain decomposition method for Helmholtz equations in unbounded domain. *SIAM Journal on Numerical Analysis*, 51(4):2331–2356, 2013.
- [11] S. Cools and W. Vanroose. Local Fourier analysis of the complex shifted Laplacian preconditioner for Helmholtz problems. *Numerical Linear Algebra with Applications*, 20(4):575–597, 2013.
- [12] B. Després. Décomposition de domaine et problème de Helmholtz. *Comptes rendus de l’Académie des sciences. Série 1, Mathématique*, 311:313–316, 1990.
- [13] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear. *ACM Trans. Math. Softw.*, 9(3):302–325, September 1983.
- [14] B. Engquist and L. Ying. Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers. *Multiscale Modeling & Simulation*, 9(2):686–710, 2011.
- [15] Y. A. Erlangga. Advances in iterative methods and preconditioners for the Helmholtz equation. *Archives of Computational Methods in Engineering*, 15(1):37–66, 2008.
- [16] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM Journal on Scientific Computing*, 27(4):1471–1492, 2006.
- [17] O. G. Ernst and M. J. Gander. Why it is difficult to solve Helmholtz problems with classical iterative methods. In Ivan G. Graham, Thomas Y. Hou, Omar Lakkis, and Robert Scheichl, editors, *Numerical Analysis of Multiscale Problems*, volume 83 of *Lecture Notes in Computational Science and Engineering*, pages 325–363. Springer Berlin Heidelberg, 2012.
- [18] C. Farhat, I. Harari, and L. P. Franca. The discontinuous enrichment method. *Computer Methods in Applied Mechanics and Engineering*, 190(48):6455–6479, 2001.
- [19] M. J. Gander and F. Nataf. AILU for Helmholtz problems: A new preconditioner based on the analytic parabolic factorization. *Journal of Computational Acoustics*, 09(04):1499–1506, 2001.
- [20] M.J. Gander, I.G. Graham, and E.A. Spence. Applying GMRES to the Helmholtz equation with shifted Laplacian preconditioning: what is the largest shift for which wavenumber-independent convergence is guaranteed? *Numerische Mathematik*, pages 1–48, 2015.
- [21] A. Gillman, A.H. Barnett, and P.G. Martinsson. A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media. *BIT Numerical Mathematics*, pages 1–30, 2014.
- [22] C. J. Gittelsohn, R. Hiptmair, and I. Perugia. Plane wave discontinuous Galerkin methods: Analysis of the h-version. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43:297–331, 3 2009.
- [23] R. Hiptmair, A. Moiola, and I. Perugia. Plane wave discontinuous Galerkin methods for the 2d Helmholtz equation: Analysis of the p-version. *SIAM Journal on Numerical Analysis*, 49(1):264–284, 2011.
- [24] R. Hiptmair, A. Moiola, and I. Perugia. A survey of Trefftz methods for the Helmholtz equation. *ArXiv e-prints*, 2015.
- [25] S. Johnson. Notes on perfectly matched layers (PMLs), March 2010.
- [26] Y. Li, H. Yang, E. Martin, K. Ho, and L. Ying. Butterfly factorization. *ArXiv e-prints*, 2015.
- [27] F. Liu and L. Ying. Additive sweeping preconditioner for the Helmholtz equation. *ArXiv e-prints*, 2015.
- [28] F. Liu and L. Ying. Recursive sweeping preconditioner for the 3D Helmholtz equation. *ArXiv e-prints*, 2015.

- [29] S. Luo, J. Qian, and R. Burridge. Fast Huygens sweeping methods for Helmholtz equations in inhomogeneous media in the high frequency regime. *Journal of Computational Physics*, 270(0):378–401, 2014.
- [30] G. Martin, R. Wiley, and K. Marfurt. An elastic upgrade for Marmousi. *The Leading Edge, Society for Exploration Geophysics*, 25, 2006.
- [31] A. Moiola, R. Hiptmair, and I. Perugia. Plane wave approximation of homogeneous Helmholtz solutions. *Zeitschrift für angewandte Mathematik und Physik*, 62(5):809–837, 2011.
- [32] A. Moiola and E. Spence. Is the Helmholtz equation really sign-indefinite? *SIAM Review*, 56(2):274–312, 2014.
- [33] P. Monk and D.-Q. Wang. A least-squares method for the Helmholtz equation. *Computer Methods in Applied Mechanics and Engineering*, 175(12):121–136, 1999.
- [34] J. Poulson, B. Engquist, S. Li, and L. Ying. A parallel sweeping preconditioner for heterogeneous 3D Helmholtz equations. *SIAM Journal on Scientific Computing*, 35(3):C194–C212, 2013.
- [35] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov. A distributed-memory package for dense Hierarchically Semi-Separable matrix computations using randomization. *ArXiv e-prints*, 2015.
- [36] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.
- [37] A. H. Sheikh, D. Lahaye, and C. Vuik. On the convergence of shifted Laplace preconditioner combined with multilevel deflation. *Numerical Linear Algebra with Applications*, 20(4):645–662, 2013.
- [38] C. Stolk. A rapidly converging domain decomposition method for the Helmholtz equation. *Journal of Computational Physics*, 241(0):240–252, 2013.
- [39] C. C. Stolk. An improved sweeping domain decomposition preconditioner for the Helmholtz equation. *ArXiv e-prints*, 2015.
- [40] A. Vion and C. Geuzaine. Double sweep preconditioner for optimized Schwarz methods applied to the Helmholtz problem. *Journal of Computational Physics*, 266(0):171–190, 2014.
- [41] S. Wang, X. S. Li, Xia J., Y. Situ, and M. V. de Hoop. Efficient scalable algorithms for solving dense linear systems with hierarchically semiseparable structures. *SIAM Journal on Scientific Computing*, 35(6):C519–C544, 2013.
- [42] X. Wang and W. W. Symes. Harmonic coordinate finite element method for acoustic waves. In *SEG Technical Program Expanded Abstracts 2012*, pages 1–5.
- [43] L. Zepeda-Núñez. *Fast and scalable solvers for the Helmholtz equation*. PhD thesis, Massachusetts Institute of Technology, Cambridge MA, USA, 2015.
- [44] L. Zepeda-Núñez and L. Demanet. The method of polarized traces for the 2D Helmholtz equation. *ArXiv e-prints*, 2014.
- [45] L. Zepeda-Núñez and L. Demanet. A short note on the nested-sweep polarized traces method for the 2D helmholtz equation. *ArXiv e-prints*, 2015.